분류된 클래스 큐를 이용한 실시간 데이터베이스 시스템의 트랜잭션 관리기

김 경 배† · 배 해 영††

요 약

본 논문에서는 트랜잭션의 예측성과 성능을 향상시키기 위한 새로운 우선순위 할당 기법과 동시성제어 기법을 제안한다. 본 논문에서 우선순위 할당 기법으로 제안한 분류된 우선순위 할당 기법은 EDF기법의 단점을 클래스와 버켓을 이용하여 해결하였고, 실시간 트랜잭션뿐만 아니라 시분할 트랜잭션까지 효과적으로 처리할 수 있다. 또한, 동시성제어 기법으로 제안된 로크를 이용한 조건부 낙관적인 동시성제어 기법은 예측성의 향상을 위해 낙관적인 기법을 사용하였으며, 시스템 자원의 낭비를 막기 위해 트랜잭션의 우선순위와 수행시간 동안 사용하는 데이터의 양을 고려하여 트랜잭션의 충돌을 해결하였다.

A Transaction Manager for Real-Time Database Systems Using Classified Queue

Gyoung-Bae Kim† · Hae-Young Bae††

ABSTRACT

In this paper, a new priority assignment policy and concurrency control for improvement of transaction predictability and performance are proposed. We present a new priority assignment algorithm called classified priority assignment(CPA), which solves the defects of Earliest Deadline First(EDF) by using class and bucket, and deals with real-time transaction and time-sharing transaction effectively. Also, we present a new concurrency control policy called conditional optimistic concurrency control using lock. It uses optimistic concurrency control for improvement of predictability, and solves transaction conflict by considering priority and execution time of transaction to waste less system resource.

1. 서 론

실시간 데이터베이스 시스템(Real-Time Database Systems)은 트랜잭션이 종료시한(deadline)과 같은 시간적인 제약조건(timing constraints)을 갖는 시스템을 말한다[12,14]. 따라서 실시간 데이터베이스 시스템의 정확성은 논리적인 결과뿐만 아니라 결과가 생성된 시간에 의존하며, 실시간 트랜잭션은 종료시한을 넘기기 전에 완료될 수 있도록 스케줄링 되어야 한다.

기존 데이터베이스 시스템에서 트랜잭션 스케줄링 목적은 트랜잭션의 평균 처리 시간을 최소로 하는 것이 목적이지만, 실시간 데이터베이스 시스템에서는 종료시간 내에 처리되는 트랜잭션을 최대로 하는 것이다[15]. 따라서 실시간 데이터베이스 시스템에서는 종료시간 내에 처리되는 트랜잭션을 최대로 하는 우선순위

† 준 회 원 : 인하대학교 대학원 전자계산공학과
†† 종신회원 : 인하대학교 전자계산공학과 교수
논문접수 : 1998년 3월 12일, 심사완료 : 1998년 8월 20일

할당 기법과 동시성제어기법을 갖는 트랜잭션 스케줄러가 필수적이다.

현재 실시간 데이터베이스 시스템에서는 우선순위 할당을 긴급한 트랜잭션에 높은 우선순위를 부여하는 EDF(Earliest Deadline First)기법[11]을 많이 사용하고 있다. 그러나, EDF기법은 트랜잭션이 적당하게 적재되는 경우에는 실시간 처리에 적합하지만, 트랜잭션이 과도하게 적재되는 경우에는 종료시간을 넘게 될 트랜잭션에 높은 우선순위를 부과하여 현격한 성능 저하를 초래하게 된다[5,6,12]. 실시간 데이터베이스 시스템을 위한 동시성제어기법으로는 두 단계 로크 기법[3]과 낙관적인 동시성제어 기법[10]이 연구되고 있다. 그러나 두 단계 로크 기법은 데이터를 로크하는 시간을 예측할 수 없으므로 예측성의 부족을 야기하고, 낙관적인 동시성제어 기법은 시간적 제약 조건을 사용하지 않는 약점을 지니고 있다[4,7,10].

본 논문에서는 EDF기법의 단점을 보완하고, 실제 실시간 데이터베이스 시스템에서 발생하는 시간적인 제약 조건을 갖는 실시간 트랜잭션과 빠른 평균 응답 시간을 요구하는 시분할 트랜잭션을 동시에 효과적으로 처리할 수 있는 새로운 우선순위 할당 기법인 **"분류된 우선순위 할당(Classified Priority Assignment) 기법"**을 제안한다. 또한 *Haritsa*[4]의 로크를 이용한 낙관적인 동시성제어(Optimistic Concurrency Control using Lock)기법을 개선하여 트랜잭션의 재시작을 줄이고 자원낭비를 줄이는 **"로크를 이용한 조건부 낙관적인 동시성제어(Conditional Optimistic Concurrency Control using Lock)기법"**을 제안한다. 제안된 기법은 실시간 데이터베이스 시스템을 위한 트랜잭션 관리기의 트랜잭션 스케줄러와 동시성제어기로 구현을 하였다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로 기존의 실시간 데이터베이스 시스템에서 연구된 트랜잭션의 우선순위 할당 기법과 동시성제어 기법들을 고찰하며, 3장에서 실시간 데이터베이스 시스템을 위한 새로운 우선순위 할당 기법인 **"분류된 우선순위 할당 기법"**과 트랜잭션의 동시성 정도를 증가시킨 **"로크를 갖는 조건부 낙관적인 동시성제어 기법"**을 제안한다. 4장에서는 제안한 **"분류된 우선순위 할당 기법"**과 **"로크를 이용한 조건부 낙관적인 동시성제어 기법"**을 사용하는 실시간 트랜잭션 관리기를 구현하였으며, 5장에서 제안된 기법의 성능을 평가하고, 6장에서 결론을 맺는다.

2. 관련연구

2.1 우선순위 할당 기법

데이터와 시스템 자원에 대한 충돌을 해결하고, 트랜잭션이 지닌 시간적인 제약조건을 만족시키기 위해서 실시간 데이터베이스 시스템에서는 트랜잭션에 우선순위를 부과하는 기법을 사용한다. 이러한 우선순위 기법으로는 가장 먼저 준비된 트랜잭션에 높은 우선순위를 할당하는 기법인 FCFS(First Come First Serve)와 실시간 데이터베이스 시스템의 스케줄링에서 가장 기본적으로 이용되고 있는 기법으로 종료시한에 가장 근접한 트랜잭션에 높은 우선순위를 할당하는 EDF, 그리고 트랜잭션이 자신의 종료시한을 만족시키면서 수행을 얼마나 늦출 수 있는 가를 나타내는 슬랙(slack)을 이용하는 LS(Least Slack)를 들 수 있다.

FCFS기법은 트랜잭션의 종료시한에 관한 정보를 사용하지 않으므로 긴급한 트랜잭션이 도착한 경우에도 먼저 대기하고 있는 트랜잭션이 앞서 처리되므로 긴급한 트랜잭션에 대해서는 대응하기가 어렵다[1]. LS기법은 트랜잭션 간에 충돌이 발생할 경우, 높은 우선순위를 가진 트랜잭션이 슬랙시간 내에 처리 가능하다면 낮은 우선순위를 가진 트랜잭션을 철회하지 않고 수행함으로써 자원의 이용을 극대화할 수 있으나, 트랜잭션의 수행시간과 이를 고려한 슬랙시간을 계산하기가 어려울 뿐만 아니라 EDF 기법에 비해 부하가 크다[13]. EDF기법은 종료시한 정보를 사용하므로 긴급한 트랜잭션의 처리에 용이하고, 트랜잭션들이 적당히 적재되는 경우에는 좋은 성능을 유지하지만, 트랜잭션들이 과도하게 적재되는 경우에는 종료시한에 근접하거나 종료시한을 초과하는 트랜잭션에 높은 우선순위가 할당되는 단점을 가지고 있다[6].

2.2 동시성제어 기법

실시간 데이터베이스 시스템의 성능을 향상시키기 위해 현재까지 연구된 동시성제어 기법은 데이터 로크를 이용하는 두 단계 로킹 기법과 데이터 충돌의 발생을 낙관적으로 생각하여 데이터 로크를 사용하지 않고 트랜잭션을 수행하는 낙관적인 동시성제어 기법을 실시간 트랜잭션의 처리에 적합하게 변형한 기법으로 구분할 수 있다. [2],[3],[8]에서 수행한 성능평가의 결과에 의하면, 충돌이 적게 발생하고 트랜잭션의 길이가 짧거나 무한대의 자원이 존재하는 경우에는

낙관적인 기법이 우수하지만, 반대로 트랜잭션의 충돌이 빈번하게 발생하고 제한된 자원 하에서 트랜잭션들이 경쟁하는 경우에는 로킹 기법이 더 우수함을 보였다.

(1) 로킹 기법 기반의 동시성제어 기법

두 단계 로킹 기법을 이용한 동시성제어 기법으로는 Abbott[1]가 제안한 기법으로 충돌이 발생한 경우 높은 우선순위 트랜잭션이 데이터에 대한 로크를 획득하는 2PL-HP(2Phase Locking-High Priority) 알고리즘과 자원 낭비를 줄이기 위해서 데이터 충돌시에 포함된 낮은 우선순위 트랜잭션의 남은 수행시간을 고려하여 높은 우선순위의 트랜잭션을 종료시한까지 완료를 지연하는 2PL-CR(2Phase Locking-Conditional Restart) 알고리즘, 그리고 기존의 공유 로크와 비 공유 로크에 순서 공유(ordered sharing)라는 새로운 로크를 첨가하여 트랜잭션의 블로킹 수를 줄이는 Agrawal[3]의 2PL-OS(2Phase Locking-Ordered Sharing) 알고리즘 등이 있다.

(2) 낙관적인 기법 기반의 동시성제어 기법

Haritsa[4]는 기존의 낙관적인 동시성제어 기법에 시간적인 제약 조건을 포함하는 동시성제어 기법으로 높은 우선순위의 트랜잭션에 종료시한을 만족할 수 있는 기회를 주기 위해 타당성 검증단계에서 높은 우선순위의 트랜잭션과 데이터 충돌이 발생한 경우 낮은 우선순위 트랜잭션이 기다리게 하는 OPT-WAIT 기법과 낮은 우선순위의 트랜잭션의 빈번한 철회를 감소시키기 위해 OPT-WAIT기법을 수정하여 데이터의 충돌에 포함된 트랜잭션들이 50%이상이 높은 우선순위 트랜잭션인 경우에만 OPT-WAIT처럼 대기를 하고, 50% 미만인 경우에는 낮은 우선순위 트랜잭션이라도 트랜잭션을 완료시키는 WAIT-50을 제안하였다.

(3) 로크를 이용한 낙관적인 동시성제어 기법

Huang[7]이 낙관적인 동시성제어 기법의 정확성을 보장하기 위하여 로크 기법을 사용한 방법이다. 이 기법은 R-Lock(Read Phase-Lock)와 V-Lock(Validation Phase-Lock) 두개의 로크를 사용하며 타당성 검사 트랜잭션이 임계 구역에서 V-Lock를 얻게 함으로써, 교착상태가 없고 병행성을 높일 수 있는 가능성을 갖는다. 낙관적인 동시성제어 기법을 사용하였을 때 트랜잭션의 직렬성을 보장하기 위해서는 다음 두 가지 조건을 만족시켜야 한다.

트랜잭션 T_i는 트랜잭션 T_j 이후에 발생하였다.

[조건 1] T_i의 갱신 연산이 판독 단계의 T_j에 영향을 주지 않는다.

[조건 2] T_i의 갱신 연산이 T_j의 갱신 연산에 겹쳐쓰지 않는다.

로크를 사용하는 동시성제어 기법에서는 [조건 1]을 만족시키고 [조건 2]는 임계영역을 사용하여 만족시킨다. 판독 단계에서 각 트랜잭션 T_i는 데이터 항목을 자신의 영역에 복사하고 자신의 $RS(T_i)$에 있는 데이터 항목에 대해 로크 테이블에 R-lock을 설정한다. 평가 단계에서는 평가 단계에 있는 트랜잭션 T_j는 $WS(T_j)$에 있는 모든 데이터 항목에 대해 로크 테이블에 V-lock을 설정한다. 만약 모든 데이터 항목에 대해 V-lock이 설정되면 평가 단계 트랜잭션의 갱신 집합과 다른 트랜잭션의 판독 집합과의 교집합이 없다는 것이다. 반대로 V-lock의 설정이 실패하면 [조건 1]을 위반하였으므로 $WS(T_j)$에 있는 데이터 항목이 다른 트랜잭션의 $RS(T_i)$에 속해 있다는 것을 의미한다.

3. 분류된 우선순위 실시간 트랜잭션 스케줄링 기법

실시간 데이터베이스 시스템에서 데이터 일관성 제약조건 뿐만 아니라 시간적인 제약조건을 만족시키기 위한 스케줄링을 위해서는 트랜잭션이 갖는 시간적인 제약조건을 이용한 우선순위 할당 기법과 할당된 우선순위, 사용된 데이터, 그리고 시스템 자원을 고려하여 데이터의 충돌을 해결하는 동시성제어 기법이 필요하다. 본 논문에서는 실시간 데이터베이스 시스템의 우선순위 할당 기법으로 많이 사용되는 EDF기법의 약점을 보완하여 실시간 트랜잭션과 시분할 트랜잭션을 효과적으로 처리할 수 있는 우선순위 할당 기법인 **"분류된 우선순위 할당 기법"**과 예측성을 향상시킨 **"로크를 이용한 조건부 낙관적인 동시성제어 기법"**을 제안한다.

3.1 분류된 우선순위 할당 기법

기존 실시간 데이터베이스 시스템에서는 종료시한 정보를 이용하여 종료시한에 가장 근접한 트랜잭션에 가장 높게 우선순위를 부과하여 긴급한 트랜잭션을 우선적으로 처리하는 EDF기법을 사용한다. 이 기법을 사용하면 트랜잭션을 완료하는 데 남은 시간이 가장 적은 트랜잭션에 가장 높은 우선순위를 부과하기 때문에 적정 수준의 트랜잭션이 시스템에 적재되는 경우에는 종료시한 내에 처리되는 트랜잭션의 수를 최대로

할 수 있는 장점이 있다. 그러나, EDF기법의 단점은 트랜잭션이 과도하게 적재되는 경우에 트랜잭션이 완료될 수 있는 충분한 시간이 남지 않은 트랜잭션에 높은 우선순위를 부과하기 때문에 연속적으로 트랜잭션이 종료시한을 초과할 수 있다. 또한, 실제 실시간 데이터베이스 시스템에서는 실시간 트랜잭션과 시분할 트랜잭션이 동시에 발생하게 된다. 즉, 시간적인 제약조건을 갖는 실시간 트랜잭션과 평균적인 응답속도를 최소화해야 하는 시분할 트랜잭션이 동시에 발생하게 된다. 따라서 EDF기법에 기초를 두면서 과부하 시에도 안정성이 있고, 실시간 트랜잭션뿐만 아니라 시분할 트랜잭션까지 효과적으로 처리할 수 있는 우선순위 할당 기법이 필요하다.

분류된 우선순위 할당 기법은 EDF기법을 발전시켜 트랜잭션이 과도하게 적재되는 경우에 발생하는 EDF 기법의 단점을 해결하고, 실시간 트랜잭션과 시분할 트랜잭션을 동시에 효과적으로 처리할 수 있도록 클래스 개념을 사용하였다. 분류된 우선순위 할당기법은 실시간 트랜잭션과 시분할 트랜잭션을 효과적으로 처리하기 위해서 실시간 트랜잭션 클래스와 시분할 트랜잭션 클래스로 구분을 한다. 실시간 트랜잭션 클래스에서는 트랜잭션의 종료시한을 만족할 수 있도록 우선순위를 할당하고, 시분할 트랜잭션 클래스에서는 트랜잭션의 평균 응답시간을 최소로 하기 위한 트랜잭션 스케줄링 기법을 사용한다.

```
알고리즘 Assign_Priority
BEGIN
 /* 수행중인 트랜잭션과 우선순위 비교 */
 IF ( TransType == REAL AND
   CurrentTrPriority<TransPriority)
  THEN
  /*종료시한 내에 완료 가능한가를 검사*/
   IF ( TransDeadline <
    (TransEstimateTime+CurrentTime))
    THEN
     /*수행중인 트랜잭션을 대기시키고
       새로운 트랜잭션을 수행 */
      Preempt new Transaction
   ENDIF
 ENDIF

 IF (TransType == REAL)
  THEN
   Assign_Real_Time_Transaction
  ELSE
   Assign_Time_Sharing_Transaction
 ENDIF
END /* end of main */
```

(그림 1) 우선순위 할당 알고리즘
(Fig. 1) Priority assignment algorithm

제안된 기법의 우선순위 할당 알고리즘은 (그림 1)과 같이 트랜잭션 관리기에 도착한 트랜잭션은 먼저 트랜잭션의 우선순위가 현재 수행하고 있는 트랜잭션의 우선순위보다 높은 경우에는 기존의 트랜잭션을 중지하고 새로운 트랜잭션을 선점시킨다. 반대로 낮은 경우에는 트랜잭션의 종류를 판독하고 실시간 트랜잭션인 경우에는 실시간 트랜잭션 클래스 큐에 트랜잭션의 정보에 따라 초기 우선순위를 할당받지만, 시분할 트랜잭션인 경우에는 시분할 트랜잭션 클래스 큐에 할당을 받는다.

(1) 실시간 트랜잭션 클래스

실시간 트랜잭션을 처리하기 위한 클래스이다. 실시간 트랜잭션 클래스는 다시 4개의 서브 클래스로 구분이 되고, 각각의 서브 클래스는 32개의 버켓을 갖는다. 각 버켓은 트랜잭션의 종료시한 초과여부에 따라 종료시한 내에 수행 가능한 트랜잭션이 위치하는 HIT 큐와 이미 종료시한을 초과한 트랜잭션이 대기하는 MISS 큐로 구분되어 있다. 실시간 트랜잭션의 처리 알고리즘은 (그림 2)와 같이 실시간 트랜잭션 큐로 들어온 실시간 트랜잭션은 트랜잭션이 갖는 우선순위에 따라 해당 서브 클래스에 할당되고 트랜잭션이 갖는 가중치에 따라 해당 서브 클래스 내의 버켓이 정해진다. 또한 각 트랜잭션의 종료시한 정보를 이용하여 이미 종료시한을 초과한 트랜잭션인 경우에는 해당 버켓의 MISS 큐에 할당을 하지만 종료시한을 초과하지 않은 경우에는 해당 버켓의 HIT 큐에 트랜잭션을 위치시킨다.

```
알고리즘 Assign_Realtime_Transaction
BEGIN
 /*해당 부클래스와 버켓 결정*/
 SubClass=(TransPriority/NoSubClass)
 Bucket =TransValue
 /*트랜잭션 종료시한 초과여부 검사*/
 IF(CurrentTime>=TransDeadline)
   THEN
    set OverDeadline
   ELSE
    reset OverDeadline
 ENDIF
 /*트랜잭션을 해당 큐에 삽입*/
 Insert Real-time Transaction
END
```

(그림 2) 실시간 트랜잭션의 우선순위 할당 알고리즘
(Fig. 2) Priority assignment algorithm of real-time transaction

(2) 시분할 트랜잭션

시분할 트랜잭션은 트랜잭션의 평균 응답 시간을 최소로 하는 것이 목적이다. 따라서 시분할 클래스 큐에 도착한 시분할 트랜잭션은 (그림 3)의 알고리즘과 같이 트랜잭션의 실행 예측 시간을 기준으로 해당 버켓을 할당받는다. 즉, 도착한 시분할 트랜잭션은 트랜잭션을 수행하는 데 예측된 실행시간을 기준으로 가장 짧은 것을 트랜잭션 큐의 위쪽에 위치시키고, 트랜잭션의 실행시간이 긴 경우에는 큐의 아래쪽에 위치하게 한다. 이 할당 방법을 사용하게 되면 롱 트랜잭션의 경우에 트랜잭션이 수행을 위해 많은 시간을 대기하는 문제점이 발생할 수 있으나, 트랜잭션의 수행시간이 짧은 트랜잭션을 우선적으로 실행하기 때문에 시스템 전체적으로 많은 양의 시분할 트랜잭션을 처리할 수 있기 때문에 전체 트랜잭션의 평균 응답 시간을 최소로 할 수 있다.

```
알고리즘 Assign Time Sharing Transaction
BEGIN
 /*해당 서브 클래스와 버켓 결정*/
 SubClass= (TransEstimateTime
            / NoSubClass)
 Bucket=TransEstimateTime
 /*트랜잭션을 해당 큐에 삽입*/
 Insert Time Sharing Transaction
 /*트랜잭션의 남은 수행시간을 기준
 적게 남은 트랜잭션을 큐 위쪽 배치*/
END
```

(그림 3) 시분할 트랜잭션의 우선순위할당 알고리즘
(Fig. 3) Priority assignment algorithm of time sharing transaction

3.2 로크를 이용한 조건부 낙관적인 동시성제어 기법

실시간 데이터베이스 시스템의 동시성제어 기법으로 기존에 안정성이 인정되는 로킹 기법보다 낙관적인 기법에 대한 연구가 활발히 전개되고 있다. 기존의 데이터베이스 시스템을 실시간 트랜잭션 처리를 위해 이용하는 데 부적합한 이유중의 하나가 바로 예측성의 부족이라 할 수 있다. 즉, 로킹 기법을 사용하게 되면 데이터를 로크하는 시간을 예측할 수 없다. 또한 교착상태의 발생과 우선순위 역행에 대한 해결책이 있어야 한다. 따라서 본 연구에서 제안하는 로크를 이용한 조건부 낙관적인 동시성제어 기법은 예측성을 향상시키기 위해서 낙관적인 기법을 사용하여 구현의 정확성을 보장하였고, 우선순위가 낮은 트랜잭션의 재수행 수를 줄여 시스템 자원의 낭비를 줄였다.

(1) 로크 모드와 양립성 함수

각 트랜잭션 Ti는 트랜잭션이 판독연산을 수행하는 데이터에 대한 판독 집합인 RS(Ti)와 기록연산을 수행하는 데이터에 대한 기록 집합인 WS(Ti)를 유지한다. 구현의 정확성을 위해서 로크는 판독 단계 로크(R-Lock: read phase lock)와 타당성 단계 로크(V-Lock: validation phase lock)가 사용이 된다. 이 두가지 모드의 양립성 함수는 <표 1>과 같다.

<표 1> 로크의 양립성 함수
<Table 1> Lock compatiblity matrix

LR \ LH	R-Lock	V-Lock
R-Lock	○	Wait
V-Lock	CR	CR

LH : Lock Holder LR : Lock Requester CR : Conflict Resolver

만약 트랜잭션이 현재 V-Lock이 설정되어 있는 데이터에 대해 R-Lock을 설정하면, 트랜잭션은 로크를 얻을 때까지 대기한다. 반면, 현재 V-Lock나 R-Lock가 설정된 데이터에 대해 V-Lock을 설정하려면 트랜잭션들 간의 충돌이 발생한 것이므로 충돌 해결 기법을 사용하여 어떤 로크를 설정할 것인지를 결정한다. 동시성제어 알고리즘은 (그림 4)와 같다

```
알고리즘 Read Phase of Ti
/*동시성제어의 판독 단계에서의 알고리즘
 Ti : 타당성 단계에 도달한 트랜잭션*/
BEGIN
 FOR(all data object to be read or written)
   place it in RS(Ti) or WS(Ti)
   set R-lock in Lock Table
 ENDFOR
END

알고리즘 Validation Phase of Ti
/*동시성제어의 타당성 단계에서의 알고리즘
 Ti : 타당성 단계에 도달한 트랜잭션
 Tj : Ti와 충돌이 발생한 트랜잭션 */
BEGIN
 VALID := true
 FOR (every data in WS(Ti))
   IF(another transaction has R-locked it)
    THEN
     VALID := false
    ELSE
     set a V-lock in Lock Table
   ENDIF
 ENDFOR
 release Tj's R-lock in Lock Table
 IF (not VALID) THEN
   invoke real-time conflict resolver
 ENDIF
 IF (VALID) THEN
   execute WRITE PHASE
 ENDIF
 release Tj's V-lock in Lock Table
END
```

(그림 4) 동시성제어 알고리즘
(Fig. 4) Concurrency control algorithm

(2) 트랜잭션 충돌 해결 기법

실시간 트랜잭션의 충돌을 해결하기 위해서 트랜잭션의 우선순위를 고려한다. 그러나 우선순위만을 고려하게 되면 완료 단계에 도달한 낮은 우선순위의 트랜잭션까지 재 수행하여 자원의 낭비를 초래한다. 특히, 낙관적인 기법을 이용하는 동시성제어 기법에서는 트랜잭션의 완료단계에서 취소되는 경우 심한 자원의 낭비를 초래한다. 따라서 시스템 자원의 낭비를 최소화하면서 트랜잭션의 시간적 제약 조건을 만족시킬 수 있는 충돌 해결 기법이 필요하다.

로크를 이용한 조건부 낙관적인 동시성제어 기법에서는 트랜잭션의 우선순위와 종료시한을 일차적으로 고려한다. 또한 자원의 낭비를 방지하기 위해서 트랜잭션의 수행시간과 데이터의 양을 고려한다. 즉, 충돌된 트랜잭션의 우선순위가 높은 경우에는 낮은 트랜잭션을 취소하지만, 트랜잭션의 무조건적인 철회로 인한 자원의 낭비를 막기 위해 종료시한까지 트랜잭션의 완료를 연기하여 우선순위가 낮은 트랜잭션이 완료될 수 있는 기회를 준다. 트랜잭션이 종료시한에 도달하면 상위의 트랜잭션은 충돌한 우선순위가 낮은 트랜잭션을 재 수행시키고 트랜잭션을 완료한다. 또한, 낮은 우선순위의 트랜잭션은 자신이 참조하는 데이터 양과 충돌한 높은 우선순위의 비율을 측정하여 트랜잭션의 재시작 여부를 결정한다. 전체 트랜잭션들이 참조하는 데이터 양의 비인 D(Ti)의 값과 충돌이 발생한 트랜잭션 중에서 우선순위가 높은 트랜잭션의 개수인 HPT(Ti)의 값을 비교하여 결정한다.

$$HPT(Ti) = \frac{\text{우선순위가 높은 트랜잭션 수}}{\text{충돌한 전체 트랜잭션 수}}$$

$$D(Ti) = \frac{\text{Ti가 참조한 데이터 양}}{\text{전체 트랜잭션이 참조하는 데이터 양}}$$

트랜잭션이 수행되면, 각 트랜잭션은 판독 연산 데이터와 기록 연산 데이터의 집합을 기록하게 되고, 사용하는 모든 데이터에 대해 판독 로크를 동시성제어기에 요청한다. 동시성제어기는 요청된 데이터에 대해 로크 테이블 속에 테이블 단위의 로크는 테이블에, 페이지 단위의 로크는 페이지에, 튜플 단위의 로크는 튜플에 해당 트랜잭션을 첨가한다. 트랜잭션이 검증 단계에 이르게 되면 동시성제어기는 트랜잭션의 기록 연산 집합에 있는 데이터에 대한 로크를 타당성 로크로 변경하게 된다. 트랜잭션간에 데이터의 충돌이 발생하지 않으면 정상적으로 트랜잭션은 종료가 되지만, 충돌이 발생하게 되면 (그림 5)의 충돌 해결 알고리즘으로 해결한다.

```
알고리즘 Conflict Resolver
BEGIN
 /*우선순위가 높은 트랜잭션*/
 IF (TransPriority==MaxTransaction)
  THEN
        wait during Transaction Deadline
   /*우선순위 높은 트랜잭션을 완료후
    충돌한 트랜잭션을 재수행*/
    Commit_High_Transaction
    restart transaction in conflict set
 ENDIF
 /*상위의 우선순위 트랜잭션과 충돌*/
 IF (conflict with high priority transaction
   AND not over Transaction Deadline)
  THEN
    wait during Transaction Deadline
    /*충돌한 트랜잭션 정보를 이용하여
    완료 또는 재수행 여부 결정*/
    Commit_Low_Transaction
 ENDIF
END
```

(그림 5) 충돌 해결 알고리즘
(Fig. 5) Conflict resolve algorithm

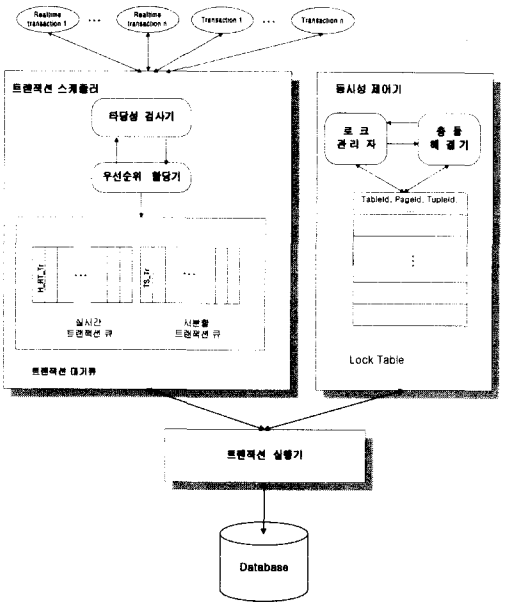
4. 실시간 트랜잭션 관리기의 설계 및 구현

본 장에서는 분류된 우선순위 할당기법과 낙관적인 동시성제어 기법을 사용하는 실시간 트랜잭션 관리기를 설계하고 구현한다. 구현된 실시간 트랜잭션 관리기는 실시간 스케줄러와 동시성제어기로 구성된다.

4.1 실시간 트랜잭션 관리기의 구조

구현된 실시간 데이터베이스 시스템을 위한 트랜잭션 관리기는 발생된 트랜잭션의 스케줄링을 담당하는 실시간 트랜잭션 스케줄러와 트랜잭션이 요구한 데이터 항목 사이의 충돌을 해결하기 위한 실시간 동시성제어기로 구성되어 있다. 실시간 트랜잭션 스케줄러는 3장에서 제안한 분류된 우선순위 할당 기법을 사용하며 트랜잭션이 종료시한을 초과하였는지를 검사하는 타당성 검사기(feasible tester)와 트랜잭션의 우선순위를 할당하는 위선순위 할당기, 그리고 대기상태의 트랜잭션들이 대기하는 트랜잭션 대기 큐(waiting queue)

로 구성되어 있다. 실시간 동시성제어기는 3장에서 제안한 동시성제어 기법인 로크를 이용한 낙관적인 동시성제어 기법을 사용하며, 데이터에 대한 로크를 관리하는 로크 관리자(lock manager)와 트랜잭션간에 발생되는 데이터 충돌을 해결하는 충돌 해결기(conflict resolver)와 로크 상태의 데이터 정보가 보관되어 있는 로크 테이블(lock table)로 구성되어 있으며, 그 구조는 (그림 6)과 같다.



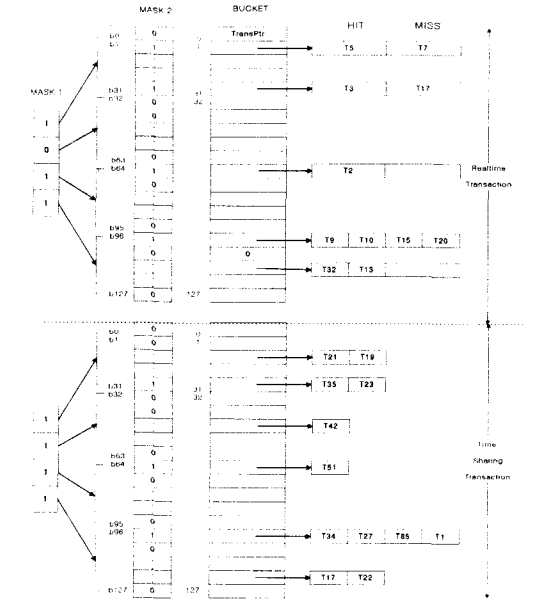
(그림 6) 트랜잭션 관리기의 구조
(Fig. 6) Architecture of transaction manager

4.2 실시간 트랜잭션 스케줄러의 설계 및 구현

실시간 트랜잭션 스케줄러는 2장에서 제안한 분류된 우선순위 할당 기법을 사용한다.

(1) 트랜잭션 대기 큐

트랜잭션이 우선순위를 할당받아 대기하는 대기 큐는 실시간 트랜잭션과 시분할 트랜잭션을 처리할 수 있도록 실시간 트랜잭션 클래스와 시분할 트랜잭션 클래스로 구분이 된다. 두개의 클래스는 트랜잭션의 구분을 위해 4개의 서브 클래스로 구분이 되며 각 클래스는 32개의 버켓으로 구분이 된다. 실시간 트랜잭션 클래스의 버켓은 다시 HIT 큐와 MISS 큐로 구분이 되며, HIT큐에는 종료시한을 초과하지 않은 트랜잭션이 위치하고 MISS 큐에는 종료시한을 초과한 트랜잭션이 위치한다. 구현된 트랜잭션 대기 큐의 구조는 (그림 7)과 같다.



(그림 7) 트랜잭션 대기 큐의 구조
(Fig. 7) Architecture of transaction waiting queue

대기 큐는 연산속도를 증가시키기 위해 두개의 비트 마스크를 두었다. mask1은 실시간 트랜잭션과 시분할 트랜잭션을 구분하고, mask2는 8개의 sub class내에서 트랜잭션의 해당 버켓 위치를 결정하기 위해서 사용한다. mask1은 8개의 비트로 구성되어 있으며, 상위 4개 비트는 실시간 트랜잭션을 위해서 하위 4개 비트는 시분할 트랜잭션을 위해서 사용이 된다. mask2는 32 * 8 = 256개의 비트로 구성되어 있고, 각 비트의 값은 0과 1을 가질 수 있으며 해당 버켓에 트랜잭션이 존재하는 가를 나타낸다. mask1의 한 비트는 mask2의 32 비트에 트랜잭션이 존재하는 가를 나타내고, mask2의 각각의 비트는 해당 버켓에 트랜잭션이 존재하는 가를 나타낸다. (그림 7)에서 트랜잭션 T5와 T7이 실시간 트랜잭션의 2번째 버켓에 위치함으로 mask1의 첫 번째 비트와 mask2의 두 번째 비트 b2가 1로 설정이 되어 있다. T5는 종료시간 내에 처리가 가능한 트랜잭션으로 HIT 큐에 위치하였고, T7은 종료시간 내에 처리가 불가능한 트랜잭션으로 MISS 큐에 위치하였다. 스케줄러는 지금 수행중인 트랜잭션이 완료되면

지명 큐의 가장 상위에 있는 트랜잭션인 T5를 수행한다. 시분할 트랜잭션에서는 T21과 T19가 동일한 수행시간이 남아 있으므로 동일한 버켓에 할당이 되어 실행을 대기하고 있다.

트랜잭션의 처리가 요청되면 타당성 검사기는 현재 수행되고 있는 트랜잭션의 우선순위를 비교하여 우선순위가 높으면 수행중인 트랜잭션을 대기시키고 요청된 트랜잭션을 선점하여 수행을 한다. 그 밖의 경우에는 트랜잭션이 종료시한을 이미 초과하였는 가를 검사하고, 실시간 트랜잭션인 경우에는 실시간 트랜잭션 큐에 트랜잭션의 가중치와 종료시한을 기준으로 지명 큐에 삽입되고, 실시간 트랜잭션이 아닌 경우에는 시분할 트랜잭션 큐에 삽입된다. 실시간 트랜잭션 큐에 트랜잭션이 삽입될 때에는 가중치에 의해 해당 큐가 먼저 결정이 되고, 큐 내에서는 트랜잭션의 종료시한 초과 여부와 종료시한의 길이에 의해 해당 버켓의 위치가 결정된다.

4.3 실시간 동시성제어기의 설계 및 구현

(1) 실시간 동시성제어기의 설계

실시간 동시성제어기는 3장에서 설명한 로크를 이용한 낙관적인 동시성제어 기법을 사용한다. 이 기법은 트랜잭션의 우선순위만을 고려하는 기존의 실시간 데이터베이스 시스템과는 달리 트랜잭션이 사용한 데이터 양과 트랜잭션 수행시간을 고려하여 충돌을 해결한다. 즉, 검증 단계에 도달한 트랜잭션은 다른 트랜잭션과의 충돌을 검사한다. 만약 트랜잭션의 충돌이 발생하지 않았으면 트랜잭션은 정상적으로 종료를 하지만, 충돌이 발생한 경우에는 충돌 해결기를 호출한다. 호출된 충돌 해결기는 해당 트랜잭션과 충돌한 트랜잭션의 우선순위뿐만 아니라 각 트랜잭션이 수행된 시간과 참조하는 데이터 양을 검사하여 해당 트랜잭션이 완료될 것인지 취소될 것인지를 결정하게 된다.

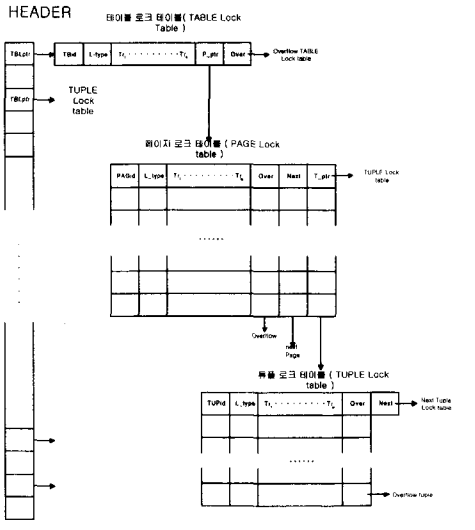
실시간 동시성제어기의 각 트랜잭션 Ti는 판독 집합인 $RS(T_i)$와 기록 집합인 $WS(T_i)$를 유지한다. 또한 수행되는 모든 트랜잭션이 공유하는 로크 테이블 LT를 유지한다. 로크는 R-lock와 V-lock 두개의 로크 모드가 사용되며, 데이터 항목에 대한 R-lock는 판독 단계에 있는 트랜잭션에 의해 결정되는 반면, V-lock는 오직 평가 단계에 있는 트랜잭션에 의해서만 결정된다.

만약 트랜잭션이 V-lock가 설정되어 있는 객체에 대해 R-lock를 설정하려면 트랜잭션은 로크를 얻을 때까지 기다려야 한다. 반면, V-lock나 R-lock가 설정되어 있는 객체에 대해 V-lock를 설정하려고 한다면 충돌 해결 기법을 사용하여 어떤 로크를 설정할 것인지를 결정한다.

판독 단계에서 각 트랜잭션 T_i는 데이터 항목을 자신의 영역에 복사하고 자신의 $RS(T_i)$에 있는 데이터 항목에 대해 로크 테이블에 R-lock를 설정한다. 평가 단계에서 트랜잭션 T_j는 $WS(T_j)$에 있는 모든 데이터 항목에 대해 로크 테이블에 V-lock를 설정한다. 만약 모든 데이터 항목에 대해 V-lock가 설정되면 평가 단계 트랜잭션의 갱신 집합과 다른 트랜잭션의 판독 집합과의 교집합이 없다는 것이다. 반대로 V-lock의 설정이 실패하면 트랜잭션의 충돌이 발생하였으므로 $WS(T_j)$에 있는 데이터 항목이 다른 트랜잭션의 $RS(T_i)$에 속해 있다는 것을 의미한다. 이 경우에는 데이터간의 충돌을 해결하기 위한 방법이 필요하다. 타당성 검사 단계에서 데이터 충돌이 검출되면 충돌 해결기를 호출한다. 충돌 해결기는 충돌한 트랜잭션의 우선순위, 가중치 그리고 수행된 시간을 고려하여 트랜잭션의 완료 또는 취소 여부를 결정한다.

(2) 실시간 동시성제어기의 구현

동시성제어기는 로크 관리자와 충돌 해결기, 그리고 로크 테이블로 구성되며, 로크 관리자는 트랜잭션이 사용하는 데이터를 로크 테이블에 관리하며, 그 구조는 (그림 8)과 같다.



(그림 8) 로크 테이블의 구조
(Fig. 8) Architecture of lock table

로크 테이블은 공유메모리를 사용하며 초기에 공유메모리 관리자로부터 할당받는다. 로크 테이블은 헤더 부분과 데이터베이스의 로크 단위를 제공하기 위해서 테이블, 데이터 페이지, 그리고 튜플로 구성된다. 트랜잭션이 수행되면, 각 트랜잭션은 판독 연산 데이터와 기록 연산 데이터의 집합을 유지하고, 사용하는 모든 데이터에 대해 판독로크를 동시성제어기에 요청하여 로크를 설정한다. 트랜잭션이 검증 단계에 이르게 되면 동시성제어기는 트랜잭션의 기록연산 집합에 있는 데이터에 대한 로크를 타당성 로크로 바꾸게 된다. 트랜잭션 간에 데이터의 충돌이 발생하지 않으면 정상적으로 트랜잭션은 종료가 되지만, 충돌이 발생하게 되면 충돌 해결기를 호출한다.

충돌 해결기에서는 자원의 낭비를 방지하기 위해서 트랜잭션의 완료를 종료시한까지 미루어 우선순위가 낮은 트랜잭션들이 완료될 수 있는 기회를 준다. 종료시한에 도달하였을 때, 높은 우선순위의 트랜잭션은 트랜잭션을 완료시키고, 충돌한 트랜잭션들을 재실행시킨다. 그러나 낮은 우선순위의 트랜잭션은 완료 단계에 있지 않은 충돌한 트랜잭션의 비율인 HPT(Ti)값이 충돌한 트랜잭션의 비율을 나타내는 임계 값(high priority transaction percent: Hp)보다 작고, 트랜잭션이 참조한 데이터 량의 비율인 D(Ti)가 충돌이 발생한 트랜잭션이 참조한 데이터 페이지의 양을 나타내는 임계 값(data percent: Dp)보다 크면 낮은 우선순위의 트랜잭션일지라도 완료를 한다.

5. 성능평가

본 장에서는 제안된 실시간 트랜잭션 관리기의 트랜잭션 스케줄러의 성능을 평가한다. 스케줄러는 우선순위, 트랜잭션의 종류 등을 고려하여 아래의 표와 같이 구분하여 평가를 수행하였다.

5.1 성능 평가 환경

(1) 시스템 환경

성능 평가를 위해 사용된 시스템환경은 아래의 표와 같다.

〈표 2〉 시스템 환경

〈Table 2〉 System environment

기 종	Sun Ultra -1
CPU 속도	167 Mhz
주기억 장치 용량	64 MB
하드디스크 용량	10GB
운영 체제	Sun OS 5.5.1
사용 언어	C++

(2) 트랜잭션 수행시간 계산

트랜잭션의 수행시간은 성능평가의 중요한 요소로써 다음과 같은 공식을 이용하였다. 트랜잭션 수행시간은 데이터베이스 시스템에서 사용자가 요구한 데이터베이스에 대한 연산을 수행하는 시간(Tdb)과 응용프로그램에서 데이터베이스 연산의 결과를 처리하는데 소요되는 트랜잭션 연산시간(Tprocessing)과 트랜잭션을 스케줄링하는 데 소요되는 시간(α)로 계산된다.

Texec_time = Tdb + Tprocessing + α

Texec_time: 트랜잭션의 수행시간

Tdb: 데이터베이스 연산시간

Tprocessing: 트랜잭션의 연산 소요시간

α: CPU의 스케줄링 소요시간

(3) 성능평가 변수

성능평가 변수는 <표 3>과 같다. 테이블 크기는 1Mbyte이고, 레코드의 길이는 100Byte이며, 연산에 사용되는 레코드 수는 10,000개이다. 트랜잭션의 평균 수행시간은 판독하는 시간이 21ms이고, 갱신 시간은 54ms가 소요된다. 트랜잭션의 충돌발생 시 트랜잭션의 수행여부를 결정하는 충돌 트랜잭션 비율 상수(Hp)와 트랜잭션의 데이터 참조비율 상수(Dp)는 50%를 사용하였다.

〈표 3〉 성능평가 변수

〈Table 3〉 Performance evaluation parameter

평 가 변 수	값	단위
테이블 크기(Table Size)	1,000,000	Byte
레코드 크기(Record Size)	100	Byte
레코드 수(Record Number)	10,000	개
판독트랜잭션 평균수행시간	21	ms
갱신 트랜잭션 평균수행시간	54	ms
우선순위가 높은 충돌 트랜잭션 비율 상수 (high priority transaction percent: Hp)	50	%
데이터 참조 상수 (data percent :Dp)	50	%

5.2 성능평가

본 논문의 성능평가에서는 실시간 데이터베이스 시스템의 평가 기준인 종료시한을 만족키는 트랜잭션의 비율을 측정하였다. 이를 위해 트랜잭션의 종류와 연산의 종류에 따른 네 가지 스케줄러를 평가하였다. 스케줄러 1은 트랜잭션의 우선순위와 트랜잭션의 종류

등을 구분하지 않고 트랜잭션이 도달하는 순서대로 트랜잭션을 처리하는 FCFS 스케줄러(s1)이며, 스케줄러 2는 트랜잭션의 종료시한만을 고려하여 처리하는 EDF 스케줄러(s2)이며, 스케줄러 3은 트랜잭션의 종료시한과 트랜잭션을 고려하여 분류된 큐를 사용하는 CEDF 스케줄러(s3)이며, 스케줄러 4는 트랜잭션의 종료시한과 종류를 구별하여 트랜잭션의 큐를 사용하고 특히, 하드 실시간 트랜잭션을 우선적으로 처리해 주는 CEDFHRT 스케줄러(s4)이다.

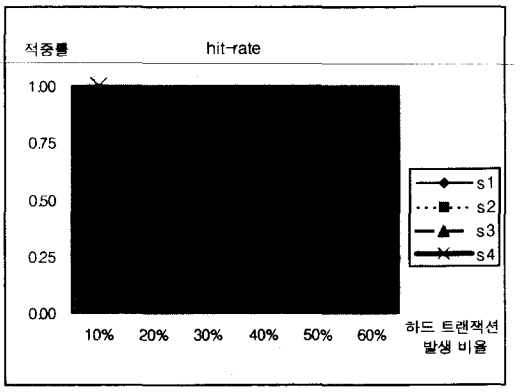
〈표 4〉 트랜잭션 스케줄러의 구분
〈Table 4〉 Type of transaction scheduler

구분 / 스케줄러	스케줄링 기법
스케줄러 1 (s1) (FCFS)	First Come First Serve
스케줄러 2 (s2) (EDF)	Earliest Deadline First
스케줄러 3 (s3) (CEDF)	Classified Earliest Deadline First
스케줄러 4 (s4) (CEDFHRT)	Classified Earliest Deadline First for Hard Real-time Transaction

(1) 하드 실시간 트랜잭션의 처리

(그림 9)는 각 스케줄러에서 실시간 시스템에서 가장 중요한 하드 트랜잭션의 성공비율을 보여주는 그래프이다. 전체 트랜잭션 중에서 하드 트랜잭션의 발생빈도에 따라 하드 실시간 트랜잭션의 성공률을 보여주고 있다. 분류된 우선순위 할당 기법을 사용하는 트랜잭션의 스케줄러가 하드 트랜잭션의 성공률이 EDF 기법이나 FCFS기법을 사용하는 트랜잭션 스케줄러 보다 월등함을 볼 수 있다.

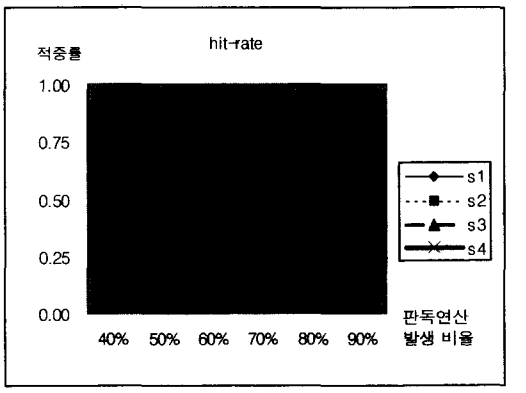
실시간 트랜잭션과 시분할 트랜잭션을 구분하는 스케줄러 s2, s3, s4는 실시간 하드 트랜잭션에 대해 높은 적중률을 보이지만, 이를 고려하지 않는 s1은 적중률이 50%이하의 처리결과를 보였다. 특히, s3, s4는 실시간 하드 트랜잭션의 발생 비율이 적은 경우에는 종료시한 내에 대부분의 하드 트랜잭션이 처리되며, 비율이 높은 경우에도 75% 이상의 성공률을 보였다. 이러한 결과는 제안된 스케줄러가 반드시 시간적인 제약조건을 만족시켜야 하는 하드 트랜잭션에 대하여 큐를 사용하여 우선적으로 처리하기 때문이다.



(그림 9) 하드 트랜잭션 비율에 따른 트랜잭션 성공률
(Fig. 9) Hit ratio of transaction by hard transaction

(2) 트랜잭션의 성공률

(그림 10)은 실시간 데이터베이스 시스템에서 발생하는 트랜잭션의 종류를 갱신 트랜잭션과 판독 트랜잭션으로 구분하였을 때 트랜잭션의 성공률을 나타낸 것이다. 즉, 판독연산의 비율이 40%에서 90%로 증가되었을 때 종료시한을 만족시키는 실시간 트랜잭션의 성공률을 표시한 것이다. 판독연산의 비율이 증가됨에 따라 적중률이 증가됨을 알 수 있다. 이는 본 연구에서 제안한 트랜잭션 관리자가 동시성제어 기법으로 낙관적인 기법을 사용하기 때문에 판독트랜잭션의 비율이 증가되면 트랜잭션간의 충돌이 적게 발생하게 되어 종료시한을 만족시키는 트랜잭션의 수가 증가됨을 의미한다.



(그림 10) 판독 트랜잭션 비율에 따른 트랜잭션 성공률
(Fig. 10) Hit ratio of transaction by read transaction

6. 결 론

본 논문에서는 실시간 데이터베이스 시스템에서 시간적인 제약조건을 만족하는 트랜잭션의 수를 최대로 할 수 있는 트랜잭션 스케줄링 기법과 동시성제어 기법을 제안하고, 이 기법을 적용하는 트랜잭션 관리기의 설계하고 구현하였다.

제안된 트랜잭션 관리기의 트랜잭션 스케줄링은 EDF기법의 단점을 보완하고 실제 실시간 데이터베이스 시스템의 응용에서 발생하는 실시간 트랜잭션과 시분할 트랜잭션을 동시에 효율적으로 처리할 수 있는 우선순위 할당 기법인 "분류된 우선순위 할당 기법"을 사용한다. 동시성제어 기법은 트랜잭션의 재시작으로 인한 자원의 낭비를 줄이기 위해 Haritsa의 로크를 이용하여 낙관적인 동시성제어 기법을 개선한 "로크를 이용한 조건부 낙관적인 동시성제어 기법"을 제안하였다. "분류된 우선순위 할당 기법"은 트랜잭션의 정보를 이용하여 트랜잭션을 실시간 트랜잭션과 시분할 트랜잭션으로 구분하여 처리할 수 있도록 하였고, 실시간 시스템의 종료시한 초과여부를 기준으로 HIT 큐와 MISS 큐로 구분하여 종료시한 내에 처리되는 트랜잭션의 수를 최대로 하였다.

또한, "로크를 이용한 조건부 낙관적인 동시성제어 기법"은 낙관적인 동시성제어 기법을 사용하여 예측성을 증대시켰으며, 자원의 낭비를 막기 위해 트랜잭션의 종료시간과 수행정보를 이용하여 병렬수행의 정도를 증가시킨다.

제안된 트랜잭션 스케줄러는 성능평가를 통해서 하드 트랜잭션 성공률이 기존의 EDF기법이나 FCFS기법을 보다 뛰어나며, 갱신 보다 판독 트랜잭션의 발생비율이 높은 실시간 시스템 환경 하에서 적합한 기법임을 보였다

참 고 문 헌

[1] R. Abbott and H. Garcia-Molina, "Scheduling Real-time Transactions," SIGMOD RECORD, ACM, Vol.17, No.1, March 1988.

[2] R. Abbott and H. Garcia-Molina, "Scheduling I/O Requests with Deadlines : a Performance Evaluation," Proceedings of 11th Real-Time Systems Symposium, IEEE, pp.113-124, 1990.

[3] D. Agrawal, A. E. Abbadi, R. Jeffers, and L. Lin "Ordered Shared Locks for Real-Time Databases," VLDB Journal Vol.4, No.1, pp.87-126, 1995.

[4] J. R. Haritsa, M. J. Carey and M. Livny, "Dynamic Real-Time Optimistic Concurrency Control," Proceedings 11th Real-Time Systems Symposium, IEEE, pp.94-103, 1990.

[5] J. R. Haritsa, M. J. Carey and M. Livny, "Value-Based Scheduling in Real-Time Database Systems," VLDB Journal, pp.117-152, 1993.

[6] J. R. Haritsa, M. Livny and M. J. Carey, "Earliest Deadline Scheduling for Real-Time Database Systems," Proceedings 12th Real-Time System Symposium, IEEE, pp.232-242, 1991.

[7] J. Huang and John A. Stankovic, "Experimentation of Real-Time Optimistic Concurrency Control Schemes," Proceedings of 17th International Conference on VLDB, pp.35-46, 1991.

[8] J. Huang, J. A. Stankovic, K. Ramamritham and D. Towsley, "On Using Priority Inheritance in Real-Time Databases," Proceedings 12th Real-Time Systems Symposium, IEEE, 1991.

[9] H. T. Kung and J. T. Robinson, "Optimistic Methods for Concurrency Control," ACM Transaction on Database Systems, Vol.6, No.2, pp. 214-226, 1981.

[10] Y. Lin and S. H. Son, "Concurrency Control in Real-Time Databases by Dynamic Adjustment of Serialization Order," Proceedings of 11th Real-Time Systems Symposium, IEEE, pp.104-122, 1990.

[11] C. Liu and J. Laylabd, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of the ACM, Jan. 1973.

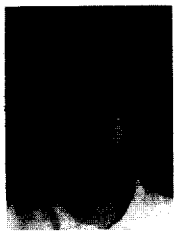
[12] B. Purimetla, R. M. Sivasankaran, K. Ramamritham, and J. A. Stankovic, "Real-Time Databases : Issues and Applications," Advances in Real-Time Systems, Prentice Hall, 1995.

[13] M. Spuri and G. C. Buttazzo, "Efficient Aperiodic Service under Earliest Deadline Scheduling," Proceedings of 15th Real-Time Systems

Symposium, IEEE, pp.2-11, 1994.

[14] Ö. Ulusoy, RESEARCH ISSUES IN REAL-TIME DATABASE SYSTEMS, Technical Report BU-CEIS-94-32, dept. of computer engineering and information science, Bilkent University, 1994.

[15] 이순조, "KORED/RT : 실시간 주기억장치 데이터베이스 관리 시스템의 설계 및 구현", 박사학위논문, 인하대학교, 1995.



김 경 배

gbkim@dbsun.cse.inha.ac.kr

1992년 인하대학교 전자계산공학과 (학사)

1994년 인하대학교 전자계산공학과 (석사)

1998년 인하대학교 전자계산공학과 (박사과정 수료)

관심분야 : 실시간시스템, 주기억장치 데이터베이스시스템, 웹GIS

배 해 영

hybae@dragon.inha.ac.kr

1974년 인하대학교 응용물리학과 (공학사)

1978년 연세대학교 전자계산공학과(공학석사)

1989년 숭실대학교 전자계산학과 (공학박사)

1985년 Univ. of Houston 객원교수

1992년~1994년 인하대학교 전자계산소장

1982년~현재 인하대학교 전자계산공학과 교수

관심분야 : 데이터베이스, 멀티미디어시스템, 지리정보시스템, 실시간시스템